Lab - Palindrome Stack

- **Due** Oct 8 by 11:59pm **Points** 25 **Submitting** a file upload **File Types** cpp
- **Available** until Oct 8 at 11:59pm

Palindrome Stack

In this lab you will create a simple program which uses a link-based stack to determine if a string is a palindrome.

Prerequisites

Chapter 8 - Lists

The Algorithm

We can use a stack to test if a string is a palindrome by first pushing each character of the string onto the stack (which results in the last character in the string at the top of the stack) then pulling the characters off the stack one-by-one and comparing them with the string. In other words, we're using the stack to create a completely backward version of the string which we then compare one character at a time to the original string.

Given a string theString and stack theStack, the pseudocode algorithm to use a stack to determine if S is a palindrome is:

```
for i = 0 through theString.length() - 1
{
   push(theString[i])
}

isPalindrome = true
for n = 0 through theString.length() - 1
{
   isPalindrome = (theStack.peek() == theString[n])
   theStack.pop()

}
```

When the algorithm completes isPalindrome will be true if the string is a palindrome, false if it is not.

The Implementation

Click here to download the starter files. Included in the starter files are:

- Node.hpp - Defines and implements a node to be used in the link-based stack.

- StackInterface.h - Defines the stack interface as seen in the textbook.
- LinkedStack.hpp - Inherits from StackInterface.h implements a link-based stack as seen in the textbook.
- main.cpp - A partially complete version of the program you'll implement in this lab.

Create a project and add all 4 files to it. The only file you should edit in the process of completing this lab is main.cpp.

Step 1

In main.cpp find the comment // STEP 1 and replace it with:

- Code that creates a pointer called stackPtr that points to a new LinkedStack of char items in free space. Hint: Remember you'll need to use the new operator to do this.

Step 2

Insert the code to prompt the user for a string and store it into inputStr. Test your program at this point. Regardless of what string you input it should say the string is a palindrome because the variable result is defaulted to true.

Step 3

We're going to implement the algorithm above, but before we do we want a way to clear a stack to make sure we start with an empty stack. Insert a void function called clearStack() which receives as a parameter which is a pointer to a stack. The function should then pop() the stack until it is empty.

Step 4

Insert the code for the function testPal() that implements the palindrome testing algorithm above. testPal should:

- Receive 2 parameters
  - One a pointer to a stack.
  - One is a string.
- Pass the stack pointer to clearStack() before using it to make sure it's empty.
- Implement the algorithm given above to determine if string is a palindrome.
- Return true if the string is a palindrome, otherwise false.

Step 5

Insert a call to testPall() passing inputStr and stackPtr as arguments. Store the return value in result.

Step 6

You should have a working program at this point. Make sure to test your program thoroughly to ensure there are no bugs.

Sample Run

When you're finished a run of your program will look something like this:

```
Enter a string and I'll tell you if it's a palindrome: dad
dad is a palindrome!
Would you like to test another string? (Yes/No): Yes

Enter a string and I'll tell you if it's a palindrome: dadda
dadda is NOT a palindrome!
Would you like to test another string? (Yes/No): No
```

Turn In

Place your name in a comment at the top of main.cpp and upload it before the due date